AD-A037 311 STANFORD RESEARCH INST MENLO PARK CALIF F/G 5/1
ACS.1: AN EXPERIMENTAL MANAGEMENT TOOL.(U)
1976 M C PEASE N00014-71-C-0210
UNCLASSIFIED SRI-TR-13 NL

| OF |
AD
A037311

END
DATE
FILMED
4-77

ADA037311

(11)

Technical Report 13

ACS.1: AN EXPERIMENTAL MANAGEMENT TOOL

By: MARSHALL C. PEASE

Prepared for:

OFFICE OF NAVAL RESEARCH
DEPARTMENT OF THE NAVY
ARLINGTON, VIRGINIA 22217

Contract Monitor: MARVIN DENICOFF, PROGRAM DIRECTOR
INFORMATION SYSTEMS BRANCH

CONTRACT N00014-71-C-0210

SRI Project 1031

D D C
RECEIVED
MAR 25 1977
C

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		*READ INSTRUCTIONS BEFORE COMPLETING FORM*
1. REPORT NUMBER 9 Technical Report No. 13	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ACS.1: AN EXPERIMENTAL MANAGEMENT TOOL. 6		5. TYPE OF REPORT & PERIOD COVERED 14 SRI-TR-13
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Marshall C. Pease 10 12 33p.		8. CONTRACT OR GRANT NUMBER(s) 15 N00014-71-C-0210
9. PERFORMING ORGANIZATION NAME AND ADDRESS Stanford Research Institute 333 Ravenswood Avenue Menlo Park, CA 94025		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 049 308
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research		12. REPORT DATE 11 1976 13. NO. OF PAGES
		15. SECURITY CLASS. (of this report) Unclassified
14. MONITORING AGENCY NAME & ADDRESS (if diff. from Controlling Office)		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this report) Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

knowledge-based system, process model, resource model, model, model-driven system, planning, administration, schedule maintenance, user control, user evaluation, exception monitoring, alerts, functional separation, modularity, consistency

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A research program on the design of knowledge-based inferential systems for the support of managerial decision-making processes is described. The principal objective has been to aid the manager in planning operations, in the administration and monitoring of approved plans, and in the retrospective analysis of operations. An experimental system, called ACS.1 for Automated Command System, has been demonstrated that exhibits properties that are considered essential for this goal.

19. KEY WORDS (Continued)

20 ABSTRACT (Continued)

From the manager's point of view, one of the most important features of the system is its flexibility. It is considered of prime importance that the manager have full control over the knowledge used by the system, and the way it uses it. He should be able to exercise this control, on either a permanent or ad hoc basis, with minimal attention to implementation details.

In addition, the research has concentrated on giving the manager flexible and effective means for evaluating the current and expected future state of his organization and its operational capabilities. It has also focused on giving him the capability of defining exceptional conditions, so that the system can monitor the expected future situation and alert him when his attention may be needed.

The technical issues addressed in the research include the system organization, the explicit separation of knowledge, data, and functions and their separation by responsibility; the maintenance of consistency in the system's working data as new data and plans are entered; the development of techniques for planning and for maintaining approved plans as conditions change; and the development of ways to integrate a complex set of models.

The design of the experimental system, ACS.1, is described in some detail. Viewed from the top level, the heart of the system is a collection of modules of two types called "planners" and "scheduler." These implement what are called "process" and "resource models," respectively. Planning is conceived as a process of negotiation among these modules, each handling its own defined responsibility. The system mimics the way the human organization develops, maintains, and executes plans. This facilitates the transfer of responsibility between the human organization and the system. This aspect facilitates incremental growth of the system, its adaptation to changing conditions and needs, and human intervention to handle exceptional situations.

It is believed that ACS.1 demonstrates principles and techniques that permit the development of practical systems for the support of the manager in the exercise of his operational responsibilities.

ACCESSION for
NTIS White Section ☑
DDC Buff Section ☐
UNANNOUNCED ☐
JUSTIFICATION ....
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A

CONTENTS

ACKNOWLEDGEMENTS

Much of the work described herein has been done in collaboration with Dr. Richard E. Fikes. Mr. Kazutaka Tachibana also assisted with some of the programming. Dr. Daniel Sagalowicz is currently collaborating on the project, and many of his ideas are reflected in the report.

I. INTRODUCTION AND OBJECTIVES

This paper describes the current state of a program of research on systems providing automated support for high level managers. The program seeks to apply some of the techniques of knowledge-based inference that have been developed by the artificial intelligence community, and to develop means for their application in decision-making environments.

In considering what attributes would make a decision-aiding system useful to managers, we have focused attention on three areas:

* User control of the system. The manager needs to be able to control the knowledge the system has, and how the system uses it. He needs to be able to exercise this control on either a permanent or ad hoc basis, with minimal attention to implementation details.

* User evaluation of conditions. The manager needs to know the current and expected future state of his organization and of its planned activities. The system should support this need, giving him access to the information in a form that is convenient for his purposes.

* Exception monitoring. The manager needs to be able to define potentially critical situations. The system should accept his specifications, and, when appropriate, alert him to the situation, allowing him to determine its significance.

To achieve these attributes, we are addressing five major technical issues so as to obtain design principles for implementing knowledge-based managerial support systems. These issues are:

* System organization. The dominant requirement is considered to be the need for user control of system initiative, and of the knowledge the system uses. This requirement is not likely to be met satisfactorily unless the system organization is explicitly chosen to facilitate user intervention. Design principles to meet this requirement are being studied.

* Separation of functions. The principle of separating knowledge, data and functions, and knowledge and data by type appears to be an important one for obtaining the required flexibility of use. The identification of areas of useful separation, and the development of techniques that will facilitate use of the different kinds of separation are important parts of the program.

* Maintenance of consistency in the data used by the system. Much of the data that describes expected future conditions or events is subject to change. New data can be entered without realization of their full implications. The knowledge contained in the system can be regarded as defining those implications, and so determining what changes in the existing data must be made to maintain consistency. The research program has given considerable emphasis to the development of principles and methods for applying the system's knowledge to the maintenance of consistency within its data.

* Planning and the maintenance of plans in a changing environment. Planning can be regarded as the seeking of ways to achieve a stated goal consistent with the knowledge that expresses the rules and constraints that must be satisfied and consistent with the current and expected future state of the environment. Planning is generally initiated by the addition of new goals. The need for the maintenance of plans is the result of changes in the environment. The research program is concerned with the development of principles and techniques to support this activity, which includes automatic planning and the maintenance of plans under normal conditions, and providing informational support in exceptional situations.

* The integration of a complex set of models. The knowledge used by the system is conceived as being embodied in two types of models. One type, called a "process model," defines what is meant by a process or an activity. The other type, called a "resource model," embodies the constraints that limit the use of some type of resource. Planning, and the maintenance of plans, is seen as requiring the simultaneous satisfaction of all these models as particularized by the command requirement and the data. The principles and techniques that will allow the simultaneous use of a large and complex set of models are being studied.

The research effort has focused on the manager's responsibility for developing and maintaining plans that are viable for his organization and resources within an environment that may change unexpectedly in many ways. This requires the initial formulation of tentative plans, their submission to the manager for modification or approval, and the maintenance and administration of approved plans. Administration, as the term is meant here, includes transmitting orders to execute planned tasks, and receiving notice of their completion. Maintenance includes the capability of recognizing when events or delays may force adjustments in existing plans. The research goals include, also, the development of means to support the analysis of past operations, based on the accumulated data.

The assumption that the environment is a changing one, and that its changes are not always predictable, is central to the research. The impact of changes, whether due to unexpected events or to unexpected delays in the execution of planned tasks, must be handled if the system is to be useful. However, it is also assumed that the frequency and extent of the environmental changes is not so great as to make planning useless. It is assumed that changes do occur that may force replanning, but that they occur with only moderate frequency.

In studying these goals, we have seen that major benefit can be obtained by the explicit separation of such entities as data, knowledge and computational functions. This principle has many aspects, some of which are not obvious. For example, the separation of functions from data or knowledge implies that the functions should be relatively independent of the knowledge that specifies their behavior under the particular circumstances. To the extent that this is done, the knowledge can be altered almost at will without rewriting the functions. A continuing theme of the research has been the identification of useful areas for the application of this principle of spearation, and the development of implementation techniques for use in these areas.

The relevance of the principle of separation to a managerial support system derives from the conviction that system design cannot be frozen if it is to remain responsive to managerial needs. These needs will change with time as the problems and opportunities change, as new circumstances arise, and as the organization develops. In addition, exceptional situations will arise that require special handling. If the system is to remain useful, it must be amenable to incremental growth, and to modification for unusual or changing requirements.

Another important principle is that of negotiation. The experimental system that is discussed herein is constructed so that viewed from the top level, it appears as a large number of essentially autonomous modules. Each module has a specific responsibility, either for planning certain defined tasks or for managing a particular type of resource. The evolution of a plan is the result of a process of negotiation among these modules, each being attentive to its own area of responsibility. Further, in the administration and maintenance of approved plans, each module accepts responsibi;ity for the commitments it has made, responding to new data in terms of that responsibility.

The significance of this principle of negotiation is that it mimics the human organization and its mode of operation. It is, then, easier to specify what the system should do since the specification of each module can be limited to its particular responsibility. Furthermore, these areas are similar to those for which human experts already exist, again making it easier to specify the behavior of the modules. Use of the negotiation principle also permits shifting responsibility for parts of the process between the system and the corresponding human organization. This provides a mechanism for exceptional situations for which the system has not been programmed since it allows responsibility to be switched to the human expert. Further, it makes possible incremental growth, since the more complex parts of the planning process can be handled by humans until the corresponding modules have been programmed and verified.

To study the technical issues described, we have undertaken the design and implementation of an experimental system called ACS.1, for Automated Command System.* Its purpose is to develop and verify principles and techniques that will be useful in the construction of practical systems.

To provide a context in which to explore the technical and managerial issues pertinent to the design of a knowledge-based management support system, we have chosen to simulate the command environment of a naval air squadron. This environment is convenient for a number of reasons. First, planning is a continuing and important part of the operation. Second, the principal activity, flying, is repetitive in the sense that the same type of tasks must be planned, using the same types of resources, independent of the details of the flight. Third, a number of different types of resources must be coordinated, giving the planning problem a significant richness. Fourth, a considerable variety of other events can happen, such as a pilot getting sick or an aircraft requiring maintenance, that can affect both the planning process and the maintenance of an approved plan.

In the following sections, the connection of this program to to other work on automated goal-seeking behavior is discussed. The the experimental system, ACS.1, is then described at successively greater levels of detail. The discussion of the system is given mostly in terms of the chosen operational environment, that of a naval air squadron. This is done in order to illustrate the practical issues involved, but the generality of the principles and techniques should be apparent.

----
* The system was previously called SPADOR for Scheduler, Planner and Administrator of Operations and Resources. The name has been changed to reflect our conviction that the principles embodied in the system have a broader significance than would be indicated by that name.

II. BACKGROUND

The problem of planning operations has received a great deal of attention over the last several years. A number of approaches have been developed and implemented in experimental systems or languages. These include the work on language understanding by Green [1], the development of TLC by Quillian [2], the STRIPS program of Fikes and Nilsson [3], PLANNER developed by Hewitt [4], MICRO-PLANNER developed by Sussman and Winograd [5], CONNIVER by McDermott and Sussman [6], the work on understanding natural language of Winograd [7], and the procedural nets of Sacerdoti [8]. The concept of "frames" as developed by Minsky [9] and Winograd [10] must also be mentioned because of its great influence on workers in the field.

The classic planning problem considered by the referenced authors and others starts from a set of possible operations. Each operation has associated with it certain preconditions. The operation is meaningful or possible only if these preconditions are satisfied. The problem is, first, to find a sequence of operations that will attain the required goal, and, second, to find one that will do so efficiently. The problem is difficult because the tree of possible sequences can become huge, far greater than can be handled by any simple search strategy. Furthermore, as Sacerdoti has observed [8], the inefficiency, or even the impossibility, of a given branch of the tree may not be detectable until a considerable effort has been invested. Consequently, major emphasis is generally given to the development and implementation of heuristic search methods.

The planning function of ACS.1 has a different character. In it, a process is viewed as a complex of operations whose sequential structure is largely determined and stable in a given managerial environment. A partial ordering, or lattice structure [11], is used to define the process that is to be planned. We admit the possibility of alternative orderings within some subset of the tasks in a process, but this is not the dominant feature of a process; the planning process is not primarily concerned with ordering the tasks in a process.

Part of the problem addressed by the planning part of ACS.1 arises from the complexity of the interactions that can occur. These interactions are assumed to occur only through competition for certain limited resources, but many types of resources may be involved. Furthermore, each type of resource may have its own rules or constraints that determine what constitutes a conflict. For example, if the resource is a human one, such as the pilots, the planning process must account for the need for rest and food.

Another part of the problem arises because it may be necessary to complete much of the planning process before a possible conflict can be identified. For example, in planning a flight, mechanics must be assigned for preflight and postflight operations. However, the times when they will be required is not known until much of the other planning has been done. It is desirable, therefore, that the initial planning at least be done rapidly to avoid an excessive investment of effort before discovering possible conflicts.

A third part of the problem arises from the variability of the operational environment. Unexpected events will occur that change the availability of some resources. Even the tasks that are planned will not be done exactly as anticipated. This leads to issues within the planning process itself. First, it implies that a plan should contain a certain amount of slack. However, if the loading on a particular resource is heavy during some period of time, then its planning should be tight, without slack. Therefore, it is desirable to be able to change the planning strategy, depending on the loading condition. Second, the likelihood that some part of a plan may prove unworkable for various reasons raises the issue of replanning. Clearly, if much replanning is necessary, it should be limited to those parts of a plan that need it.

Thus, the aspects of the planning problem addressed by ACS.1 arise from the complexities of the processes being planned, of the interactions between plans and with other events, and of the operational environment. These complexities make the problem substantially different from those addressed in the research referenced above [1-11].

The planning problem is also substantially different from the usual scheduling problem, which is discussed in Reference 12. The classic scheduling problem is to find a suitable ordering of a set of jobs subject to some set of constraints. The suitability of an ordering may include not only that all constraints are satisfied, but also that some cost function shall be minimized, or value function maximized. The difficulty of the problem lies in the size of the tree of possible orderings, so that heuristic methods may be needed to guide the search.

In the type of application environment addressed by ACS.1, it is assumed that replanning will be needed frequently for a variety of reasons: an unexpected event, a failure to complete a task on time, or to make room for another high priority operation. Replanning can be expected to be reasonably common.

If replanning is frequent, it is unlikely that we can afford to put primary reliance on a complex scheduling algorithm. Although we may use one occasionally to establish an initial planned optimal use of the resources, most of the replanning can be expected to depend on finding a way of "patching" the current schedule to make it work without reoptimizing it. Indeed, much of the replanning is likely to be only partial, just enough to recover workability.

The research problems addressed in ACS.1 are, therefore, not those of optimal scheduling, although we can expect that it will be necessary to integrate a sophisticated scheduling procedure into ACS.1 for many applications. The immediate concern is to make the planning process function satisfactorily, and to provide for the maintenance of approved plans as the situation develops.

In the following sections, the structure of ACS.1 is described, as well as the design principles and techniques that make it responsive to managerial needs.

III. OPERATIONAL OVERVIEW

In this section, the operations of ACS.1 are described from the user's point of view. The description is given in the context of a naval air squadron. The intent is to provide a perspective from which to understand the reasons for the system's various features, no to limit the application of the principles and techniques being studied. As indicated before, we are addressing technical problems relevant to the design of systems for the support of managers in a wide class of application environments.

In the current system, planning is initiated by the commander entering a requirement to fly a mission to a specified destination at a specified time, and to leave the destination area at a specified later time. ACS.1 is required to generate a plan that will meet these requirements. Constructing a plan requires the identification of the pilot and aircraft, and the determination of start and end times for preflight servicing, fueling, arming, launch, recovery, and for postflight inspection. Maintenance and service personnel must be assigned for some of these operations. The launch and recovery facilities must also be scheduled. A plan has been constructed only when the start and end times of all tasks have been set, and all necessary resources have been assigned for the necessary intervals of time.

If a plan can be generated that satisfies all of the constraints, it is returned to the commander for his consideration. He may modify the plan, direct its acceptance, or cancel the requirement. It is important to note that ACS.1 is intended to aid him in the exercise of his responsibilities, and does not preempt his responsibilities.

Once a plan has been accepted by the commander, ACS.1 assists in its execution. As time progresses, it notifies the appropriate people or departments when tasks should be started, for example, that preflight servicing should be initiated. It checks the time of completion of the tasks, determining that the work is going according to plan. If a serious lapse from the plan is observed, it will seek to replan the operation to meet the original requirements and will report the situation to the responsible authority. Thus ACS.1 assists in the execution of the plan and monitors its continuing validity.

After the mission is completed, ACS.1 records the data in its data system. These data remains available for retrospective analysis by the commander or his staff.

ACS.1 also accepts other data that may be given to it that can have significant impact on the availability of resources. For example, if a pilot becomes sick, or an aircraft is found to need maintenance, these facts are recorded since they affect availability. In responding to a requirement to plan a mission, ACS.1 takes account of this information, using it to avoid making unrealistic assignments. In accepting other information, ACS.1 is also able to recognize its

implications for existing plans. For example, if a pilot becomes sick, ACS.1 determines if plans have been approved that use that pilot. If so, those plans have become unrealistic. If authorized, ACS.1 will replan the affected missions using other pilots. The revised plans, or the failure to replan successfully, is reported to the commander for his approval or modification. If ACS.1 has not been given replanning authority, it reports the situation to the commander, alerting him of the need for action.
action.

ACS.1, then, monitors the continuing validity of the approved plans, guarding lest they be invalidated by later events or information.

In providing the capabiliity to plan, and to aid in the administration and monitoring of approved plans, ACS.1 also provides the means through which the user can conveniently modify the rules and procedures used by the system, either permanently or temporarily. The user can suspend the system's role at any point, transferring responsibility to the corresponding human authority. He can, in other words, use the capabilities provided by ACS.1 in any way and to whatever extent that best serves the existing situation and his own needs.

IV. FUNCTIONAL OVERVIEW

In this section, the internal organization of ACS.1 is considered. Attention is limited to the system concept; details are discussed in later sections.

ACS.1 has been implemented on a PDP-10 using INTERLISP under TENEX. Figure 1 is a block diagram of the system at the conceptual level; its actual implementation is somewhat different. In particular, various functions are shared among the modules shown in Figure 1, rather than being duplicated as would be necessary for full modularity. Nevertheless, the virtual modularity permits the user to make modifications, or to change the system's operations, as if the design were literaly as given in Figure 1.

ACS.1 operates with a simulated clock in an interactive mode. This is sufficient for research purposes. Experience indicates, however, that it would not be unreasonable to expect satisfactory real-time performance on a dedicated machine, even without further optimization of its code.

The heart of the system is a collection of modules, each of which is responsible for particular planning operations, of for scheduling particular types of resources. These are virtual modules, but the apparent modularity is an important element in providing user control, as is discussed later.

The two principal types of modules are the planners and schedulers shown in Figure 1. All interactions among them, and communication to the commander or other user, are handled by messages. A plan is developed as a set of agreements between the modules reached through negotiation among them. The process is initiated by the commander entering a requirement for a plan, for example, to fly a particular mission. This activates the mission planning module and causes it to initiate requests for planning the various tasks that are required and for assigning the needed resources. The requests for planning the tasks are sent to the planners that know how to plan them. The requests for assignments go to the appropriate schedulers. The planners for the subtasks may call other planners or schedulers, so that the process may involve many levels of planning at different levels of detail. A planner, at whatever level, reports that it has a plan only when all of its component tasks have been planned and all necessary resources have been scheduled.

The knowledge used by the system is held in the planners and schedulers. Knowledge held by a planner describes the essential features of the process handled by that planner, and is called the "process model" for that process. The knowledge held by a scheduler describes the constraints on the use of a particular type of resource, such as pilots or aircraft, and so controls how that type of resource can be utilized for various processes. This knowledge is contained in what is called the "resource model" for the resource type. The

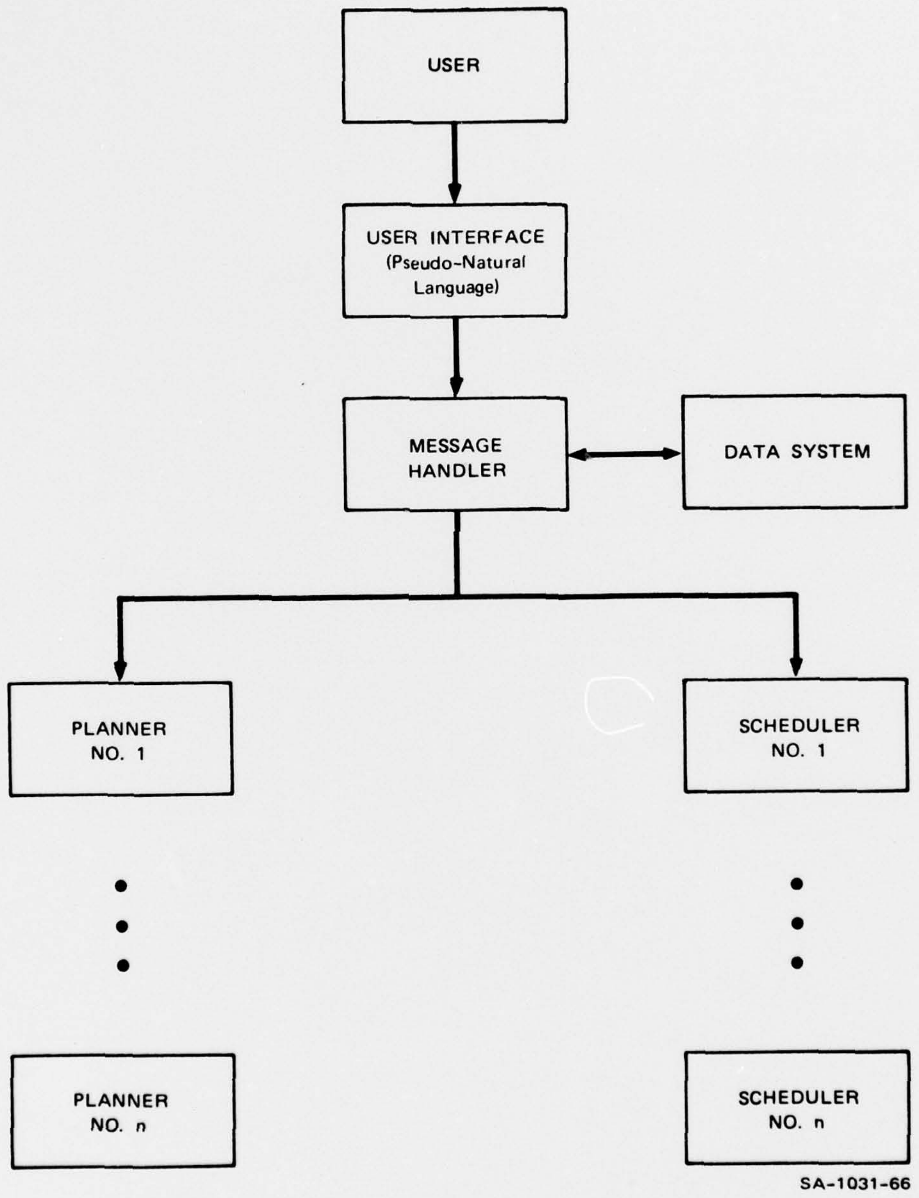

FIGURE 1 SIMPLIFIED BLOCK DIAGRAM OF ACS.1

structure and use of these models is described in later sections.

In addition to the set of planners and schedulers, there are three other major components. One is the data system that records historical information about completed operations. The second component is a pseudo-natural language front end which serves as an interface to the user.* The third component is the message handler.

A relational file system has been constructed for the data system, although it not yet been integrated into the system. It is planned that the data system will be an active module that will handle the monitoring function on its own initiative. All data entering the system will go to the data system which will determine the significance of the data versus established deadlines. If the situation requires replanning, this will be discovered in the data system and the appropriate process will be initiated.

The message handler is a central switching module that handles all communications among the planners, schedulers, the data system, and to and from the user. It is also planned that the message handler will be able to translate messages. Receiving a messgae from Module A for delivery to Module B, it will translate that message into a format that can be received by Module B. This will further decouple the modules from each other, facilitating their modification or the substitution or addition of new modules.

The central location of the message handler, and its design, are important for obtaining flexibility and facilitating command intervention. It permits having more than one planner for a given process or task, implementing alternate ways of planning the task. The user will be able to switch from one to the other, either permanently or on an ad hoc basis to meet exceptional circumstances, by changing the contents of the message handler. He can also cause it to send all messages of a given type to the terminal, allowing him to take direct control of any part of the planning process.

In summary, the structure of ACS.1 is deliberately designed to parallel the corresponding human organization. The responsibilities of the planner and scheduler modules correspond, to a large degree, to those that are likely to be assigned to departments or to specific individuals. The message handler provides a central point at which the manager can intervene to modify this separation of responsibilities within the system, or to shift responsibility between the system and the human organization.

----

* By a pseudo-natural language front end, we mean one which can accept input queries and commands in a natural language format. The formats, however, are specified, and the system uses pattern-matching, rather than syntactic and semantic analysis. The program we have used is one called LIFER that has been developed by the Artificial Intelligence Center at SRI.

V. PLANNERS

The responsibility of a planner is to plan some activity. This activity may be a top-level one, such as a mission, ordered by the manager. Or it may be a lower level activity that is a part of a higher level one, such as the preparation of an aircraft for flight.

In order to execute its responsibility, a planner must contain the necessary knowledge about the activity. This knowledge constitutes what we call a "process model." The process model for any activity, whether top level or subordinate, identifies what information is required in a plan for that activity, how that information can be obtained, and what are the applicable constraints. A given process model specifies, implicitly, a particular level of detail. The information that it seeks and uses is strictly confined to that level of detail. Any more detailed information that is needed to generate the required information will be handled by a subordinate planner using its process model.

As an example, Figure 2 shows the principal components and relationships contained in the process model for flying a mission.

The correspondence between a process model, as illustrated in Figure 2, and a PERT chart is evident. The correspondence is not complete. A PERT chart includes also the relevant deadlines, which become part of a plan that instantiates the process model.

More exactly, a process model contains the following information about the activity it models:

(1) The tasks that are components of the process being modeled.

(2) The way in which the duration of each task is to be determined for planning purposes. This may be by stored values or through specifying a call on another planner that knows how to compute the duration.

(3) Any sequential constraints between tasks, such as that certain tasks must be completed first.

(4) The identities of the planners that can plan the tasks identified in (1).

(5) The types and number of resources that must be assigned during execution of the process.

(6) The constraints that relate assignments to tasks within the process-- for example, that the assignment must be made from the beginning of one task to the end of another.

(7) The identities of the schedulers that are responsible for scheduling the needed resources.

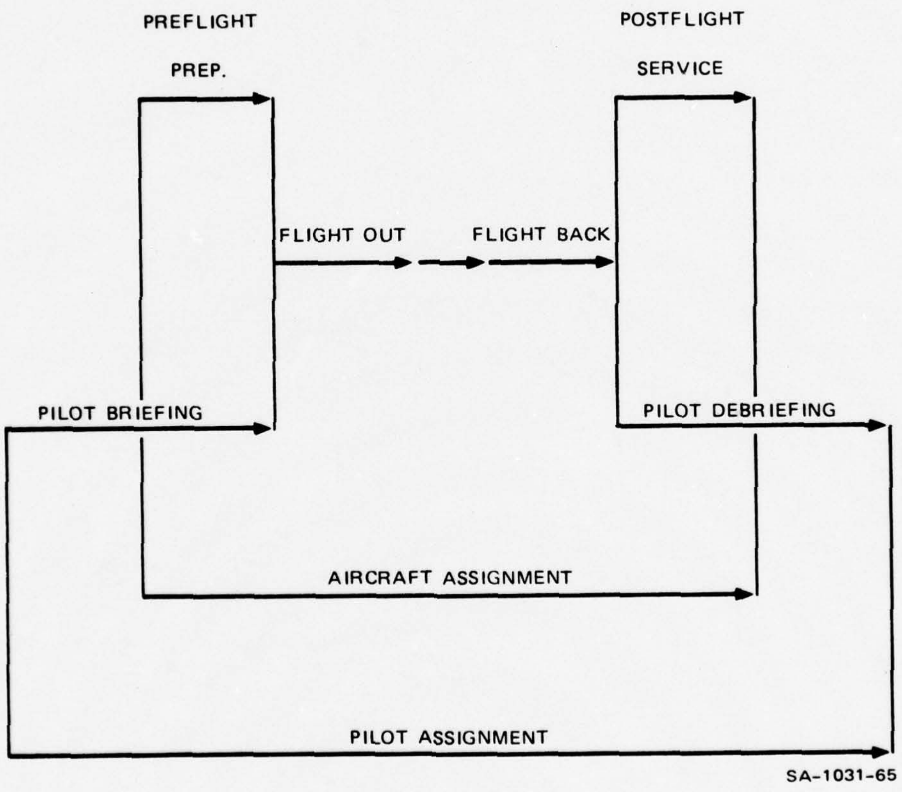

FIGURE 2   PROCESS MODEL FOR FLYING A MISSION

(8) The formats and contents of messages to schedulers and planners that will request assignments and the planning of tasks, and of the expected responses.

As an example, the process model given in Figure 2 indicates that flying a mission is composed of the tasks called preflight preparation, pilot briefing, flight out, the unnamed task at the target, flight back, postflight service, and pilot debriefing. The graph at the top indicates the constraints between these tasks, such as that the preflight preparation and pilot briefing must be completed before the start of the flight out. The model also states that an aircraft and a pilot must be assigned, the aircraft being assigned from the start of the preflight preparation to the end of the postflight service, and the pilot being assigned from the start of the pilot briefing to the end of the pilot debriefing.

The durations of the tasks before and after flight can be handled through stored values, based either on established standards or through experience. The durations of the flight out and the flight back, and thus the times of launch and recovery, must be computed by a separate planner using the ship's position, the target's location, and the airplane's performance characteristics.

The planner that plans the flight out and back can be regarded as using a more conventional type of model, one that models the geography involved and the performance characteristics of the aircraft. This illustrates the use of other types of models, which is feasible because of the modular organization of the system as a whole and the use of messages for communication within the system.

In addition, the process model will identify the planners that can plan the various tasks identified in Figure 2. For example, the process model for the preflight servicing will cover inspection, arming, fueling, and locating the aircraft on the flight deck.

It is no accident that the constraints expressed in Figure 2 all appear quite obvious. Our feeling is that, if they are not obvious, the given process model should be decomposed further into a hierarchy of models, each of which would have relatively trivial constraints. Otherwise, there is too great a danger of ambiguity, or of overlooking some important implication. Also, the decomposition into relatively simple process models enhances adaptability and flexibility since it makes it easier to understand the content of each model and its implications.

The subtasks of launch and recovery require the assignment of facilities and crews, but may not be decomposed further into smaller tasks. If not, their process models consist only of the single task and the assignment. This is a limiting case in which the process model has become structurally trivial.

The implementation of a process model is accomplished by means of an a-list, or an association list. An a-list is a list of property-value pairs without specified order. Any retrieval of a value is done associatively, using the name of the property. Also, the values may themselves be a-lists, so that the whole structure can be a tree of indefinite depth. A-lists are a convenient device that is well implemented in INTERLISP for encoding complex information.

The principle information encoded in the a-list of a process model is that listed above. Other information, such as the rules for handling priorities, can also be encoded in the a-list.

The development of a plan starts from the specification of its objective. Some timing information must be contained in the request; for example, the request may specify the time of arrival at the target, and the time of departure. The planner determines which tasks are partially but not completely specified--in this case, the flight out and the flight back. It initiates requests to supply the missing information to the planner that plans flights. The responses fills in the information about the flights out and back. These data also provide an end time for preflight preparation and pilot briefing, and a start time for postflight service and pilot debriefing. Hence these tasks are now partially specified and can be planned by the corresponding planners.

After all the tasks named in the process model for a mission have been planned, the mission planner knows the intervals over which an aircraft and a pilot must be assigned. It can then request these assignments of the schedulers identified in its process model.

The planning of the tasks in the mission by the various planners may include other processes and assignments. For example, the flight out may include the tasks of launching and the flight itself. The flight itself will be computed by another planner that knows the ship's location and the flight characteristics of the aircraft type. The launch task requires that the catapult be scheduled by the appropriate scheduler. The crew operating the catapult may also need to be scheduled. The planner that is planning the flight out will not respond to the initial request until these tasks have been planned and the needed assignments made.

In the course of generating a plan instantiating a process model, some additional complications may arise. First, there is the use of internal priorities. For example, the ideal launch time is determined by the specified time at the target. However, an earlier launch time is acceptable, providing it is not too much earlier. Hence, in requesting assignment of a catapult, the desired launch time is specified with top priority, and the beginning of the launch with lower priority. The catapult scheduler will then meet the requested times if it can, but, if not, will furnish an earlier rather than a later time, violating the requested start time, rather than the end time.

A second complication may entail external priorities, given as part of the initial requirement by the commander. This has not been implemented as yet, but is intended to be introduced when time permits. It would be used to determine when resources can be preempted from other, previously approved plans if necessary for meeting the requirements.

Finally, there is the possibility of alternate sequences. In the example of a military flight, arming and fueling can be done in either order, although not concurrently. If the planner for the preflight preparation cannot obtain a fully satisfactory plan using one sequence, it will try the other.

The encoding of the process model as a distinct data structure is an important feature in making the system amenable to managerial control. It makes it relatively easy for the manager to specify a new process model, or to change an existing one. If he wishes to introduce a new planner, either to replace an existing one or to plan some activity that has not been planned by the system before, he need only introduce the appropriate process model or models. Dialog functions for this purpose have been written.

As indicated before, it is also important that the process models are relatively simple, with complex processes being decomposed into a hierarchy of process models at successively greater levels of detail. Again, this facilitates intervention by the manager to change the process being planned or to change the scope or purpose of the system.

In addition, work is currently under way to provide for planning in context. That is, we expect to be able to hold plans on a contingent basis, from which they will be promoted to the global context for execution only when needed.

IV. SCHEDULERS

The general responsibility of a scheduler is to coordinate the use of some type of resource, such as the pilots, the aircraft, the launch catapult, or any other resource that needs to be coordinated. The word "resource" includes people, equipment, or facilities, as needed.

The principle system function served by the schedulers is to respond to requests for assignment from the various planners. To do this, each scheduler must have access to all information that bears on the future availability of its resources. In ACS.1, this information is retained by the scheduler itself in a data structure called a "scroll table," which is described later. This data structure is entirely separate from the data system of Figure 1, which records actual events rather than planned or expected future ones.

A scheduler is also required to monitor its data to recognize when existing, approved plans must be reconsidered. This requires that it be able to enforce consistency on the data given to it, as defined by the knowledge it has about its resources. For example, if an aircraft is found to be inoperable, this precludes its use until the needed maintenance is done. If it has been scheduled for a flight, then either the required maintenance must be done first, or the plan for the flight must be revised. It is the responsibility of the aircraft scheduler to recognize this fact, and to initiate the appropriate system action.

To execute its responsibilities, a scheduler must contain knowledge about the constraints that limit the use of the resources that are its concern. This knowledge is contained in what we call a "resource model." The way this knowledge is encoded and used is discussed later.

The ability of a scheduler to retain data about the future use of its resources makes it a convenient location for providing additional services to the manager. It can be expected, for example, that the manager will need to obtain various information about the expected usage of the various types of resources, retrieving either specific data or various summaries and overviews. This need is supplied by the schedulers of ACS.1 in response to queries from the user that are transmitted through the message handler.

In addition, a scheduler is a reasonable place to locate some of the alert functions that the manager is likely to need. An "alert function" is one that will inform the manager if some critical condition occurs that may require his attention. Since the scheduler holds information about the future state and availability of its resources, it has the information needed to recognize critical situations of overloading or underloading.

For example, the commander of an air squadron may specify that a certain number of pilots should be available during some critical period. The pilot scheduler has the information with which to recognize whether or not this condition is being maintained. Therefore, it should have the capability of monitoring the situation, and of alerting the commander when the requirement is being violated.

In summary, for the schedulers to execute their responsibilities, they must have the following capabilities:

* A scheduler accepts and maintains all information that affects the future availability of the resources it controls.

* The information retained by a scheduler is maintained in a self-consistent way, as specified in the rules and constraints of its resource model.

* The information is retained in a form that permits the user to obtain overviews of the expected utilization of its resources in accordance with his needs.

* Tools are provided with which the user can define critical conditions of overuse or underuse of which he needs to be alerted.

To provide these capabilities, the schedulers use a data structure called a scroll table. Table 1 is an example of the scroll table for a pilot scheduler after it has been given certain data and and has participated in planning certain missions.

TABLE 1
PILOT SCROLL TABLE

```
<MONITOR>:  SHOW THE TABLE FOR THE PILOTS FROM 0:00:00 TO 0:03:00

TABLE NAME:   PILOTS
DATE:   70001.00:00

NAME\TIME|      0:00    0:30    1:00    1:30    2:00    2:30
------------------------------------------------------------
ABLE     |      SICK    SICK    SICK    SICK    SICK    SICK
BAKER    |      AVAIL   ASG     ASG     ASG     ASG     ASG.RET
CHARLES  |      AVAIL   AWAY    AWAY    AWAY    AWAY    AVAIL
DAVIS    |      AVAIL   ASG     ASG     ASG     ASG     ASG.RET
ELLIS    |      AVAIL   AVAIL   AVAIL   AVAIL   AVAIL   AVAIL
```

Table 1 is a copy of an actual printout, except for the addition of the dividing lines. The command after "<MONITOR>:" is an example of a pseudo-natural language input that is recognized by the User Interface of Figure 1.

A scroll table is a two dimensional array of indefinite size, arranged in columns of increasing time. As shown in Table 1, the rows are assigned to the different resources being scheduled, here the pilots in the squadron. The columns represent increments of time, 30 minute intervals in this instance. The first column includes the current time. As time advances beyond the limit of the first column, that column is deleted and the second column becomes the first. The table is then said to have been "scrolled."

In Table 1, information about a given pilot at a given time is found in the "cell" that is at the intersection of the appropriate row and column. In the actual printout shown, only a small portion of the content of a given cell is exhibited, specifically a code that identifies one of a predetermined set of states for the named resource. The additional information contained in a cell can be obtained by accessing the cell directly. For example, pilot Baker at time 0:30 is shown to be in the state ASG, or assigned to a flight. Accessing the cell will recover the identification number of the mission, its start and end times, and other data that may have been included such as the flight purpose code. Similarly, at time 2:30, he is is the state ASG.RET, for assign-return, indicating a rest period following an assignment. Accessing that cell will recover not only the start and end times of this state, but also the key data on the assignment from which he is resting.

Table 1 shows the scroll table in its basic conception. It exists in this form only as a virtual entity, i.e., as one that is implied by the next more detailed level of implementation, although it can be printed out as if it actually existed. At the next level of detail, the scroll table has the form shown in Table 2, which is also a copy of a printout.

TABLE 2
CONDENSED PILOT SCROLL TABLE

```
<MONITOR>:  SHOW THE CONDENSED TABLE FOR PILOTS

PROPERTY:  STATE.NAME                   (* = AVAIL)
   TABLE:  PILOTS

NAME\TIME|      00:00    00:30    02:30    05:00    05:30    08:00
-------------------------------------------------------------------
ABLE     |      SICK     SICK     SICK     SICK     SICK     SICK
BAKER    |      *        ASG      ASG.RET  ASG.RET  *        *
CHARLES  |      *        AWAY     *        *        GONE     GONE
DAVIS    |      *        ASG      ASG.RET  ASG.RET  *        *
ELLIS    |      *        *        *        *        *        *
```

The major difference between Tables 1 and 2 is that, in the latter, the individual columns may cover an indefinite number of increments of time. For example, the second column of Table 2 covers the time period from 0:30 to 2:30. although the basic increment of time is 30 minutes. The data in that column apply to all half-hour intervals in the range. A new column is made only when it is actually needed. For example, if pilot Able were to return from sick leave at, say, 1:30, the first action would be the creation of a column whose start time was 1:30. This column would , initially, be given the same data as that contained in the column with start time 0:30. The information that pilot Able will become available at 1:30 can then be entered into the table.

Columns are created only as the need arises. Therefore, the total time covered by the scroll table is limited only by the capability of the computer to handle the numbers involved. The applicability of this device depends on the assumption, which is reasonable in the case of a scheduler, that most of the data that must be stored will concern the fairly immediate future. It allows the scroll table to cover an essentially indefinite period without requiring an excessive amount of memory.

It may be observed, in Table 2, that there is no apparent difference between the column starting at 2:30 and that starting at 5:00. The same is true of the last two columns. The creation of the extra columns was the result of making an assignment to a mission that ended at 4:50, and then cancelling that mission. These columns are, therefore, no longer needed, and could be combined into the previous columns. However, it is not worthwhile to do so; no check is made to determine if columns can be safely removed. Scrolling will eventually eliminate unnecessary columns.

As stated previously, the information printed in Tables 1 and 2 is only a small part of that actually contained in the cells of the scroll table. This is illustrated in Table 3, which gives a rundown of the available information about each pilot at the requested time, in this case 5:15. Table 3 gives, for each pilot, the start and end times of the principal actions involved, and the identification code for that action where appropriate. By "principle action" is meant the action that was initially entered, which may be different from the action indicated by the state name. For example, the start and end times, and the identification code, for Davis are those of the original assignment, although ASG.RET is the state name for the rest period following an assignment.

TABLE 3
PILOT ASSIGNMENTS

```
<MONITOR>:  WHAT IS THE ASSIGNMENT OF EACH PILOT AT 0:05:15

NAME       |     STATE      START       RETURN       ID
-----------------------------------------------------
ELLIS      |     AVAIL
DAVIS      |     ASG.RET    33          123          M3
CHARLES    |     AVAIL
BAKER      |     ASG.RET    32          122          M2
ABLE       |     SICK       0           INDEFINITE
```

Additional information can be attached to the scroll table in other ways than by entry into its cells. For example, attached to the table as a whole is a list of the missions that have been given to the scheduler, together with the key information about each. A printout derived from this list is shown in Table 4. This listing permits accessing the data about missions directly through their identification codes without having to search the table. It acts like an inverted file for that information. Note that Table 4 contains data about a cancelled mission, number M1, which no longer appears in the scroll table. This is useful since it permits the plan for that mission to be reinstated without replanning, if this can be done without conflict. It increases the flexibility of the scheduler.

TABLE 4
MISSIONS

```
<MONITOR>:  SHOW THE STATUS OF EACH MISSION

ID         |     PILOT   START   RETURN    STATUS
-----------------------------------------------
M3         |     DAVIS   33      123       SCHEDULED
M2         |     BAKER   32      122       SCHEDULED
M1         |     BAKER   31      291       CANCELLED
^L
```

If desired, it would also be possible to attach data to the rows or columns of the scroll table. For example, total flight hours for the month could be accumulated for each row and attached to it.

The format of the scroll table, either in full or in its condensed form, is a useful way of displaying key information to the user. It should be noted that any information that has been encoded into the cells of the scroll table can be displayed in this format. Hence it provides the user with the ability to obtain an overview of the expected utilization of the resources of a given type, according to whatever property he wants to see.

The structure of the scroll table is convenient for planning purposes. It permits rapid response to a request for assignment. When a request is received, the columns for the requested period can be scanned rapidly to determine which resources are available. The structure of the scroll table is also convenient for determining the effect of an input describing an unanticipated event, and thus for maintaining the consistency of the data. For example, if a given pilot goes on sick leave, the corresponding row can be scanned rapidly to determine if any plans are affected. This is not, in fact, the way consistency is maintained; the actual method is discussed later. However, the method used does depend on the ability to access data rapidly in a row over a given time period.

The scroll table is particularly useful for queries or entries that specify the time, but not the particular resource. This is generally true of requests for assignment. It may also be true of queries from a user, such as the commander, about the utilization of the resource type. The importance of these uses of the schedulers that has dictated the use of scroll tables in ACS.1.

The scroll table is currently being modified to include the possibility of entering data "in context," that is, when the data are only provisionally true, or refers to some contingency. This capability will be coordinated with the ability to plan "in context," as, for example, in constructing contingency plans. Provision is also being made to allow the user to obtain overviews, and to set alert functions, with reference to a specified context.

V. RESOURCE MODELS AND CONTROL OF THE SCHEDULERS

In the previous section, the way information is held in a scheduler, using the device of the scroll table, is discussed. In this section, we discuss the resource model that contains the knowledge used by a scheduler, and the techniques that enable the scheduler to use that knowledge.

The resource model of a scheduler contains the knowledge about the resource type being scheduled. This knowledge is used to identify the entries that can be made to its scroll table and to define the consistency relations that must be enforced.

The principal components of a resource model are the "entry types" that are to be recognized by the scheduler. These are labels that are used by the planners in requesting assignment, and by the data entry functions to identify the category of the entry desired. For example, the entry types that have been used by the pilot scheduler are: MISSION, MISSION-REST, TRAINING, SICK-LEAVE, SICK-LEAVE-RETURN, LEAVE, ATTACH and DETACH. MISSION-REST is a period that follows assignment to a flight during which the pilot is not to be assigned to another flight if it can be avoided. SICK-LEAVE-RETURN is a similar period following sick leave. Other entry types could be added easily, and probably would be required in an operational system. This set, however, provides an interesting variety of constraints.

For each entry type in the resource model, the following information is included:

* The state name that identifies the entry type--for example, ASG for missions. (Note: Only a single state name is used for each entry type.)

* The identity of any preceding or following entry type--for example, MISSION-REST as a succeeding entry type to MISSION. (Note: This permits multiple state names to describe a connected sequence of entries, each being a single entry type.)

* The "level" of the entry type, which determines if the type can displace data already present in a cell. Entry is allowed only if the level of the entry type is greater than than that of the existing data.

* A code designating the mode of entry, identifying how the entry is to be made. For example, MISSION must be entered consecutively in a single row of the scroll table over the entire period specified. TRAINING (meaning classroom training, not flight training) can be entered disconnectedly in a single row, providing the required total time is accumulated.

* The labels and format of additional information that may be included. For example, MISSION requires the designation

of an ID under the label ID-MISSION. The resource model specifies the label and, by implication, that the entry shall be listed separately and attached to the table as a whole, as illustrated in Table 4.

* The demons* that are to be set, as is discussed shortly, and the names of the functions to be used.

* If appropriate, the functions called by the demons when they are fired. These include, for example, dialog functions that give the user direct control of what the system does about the situation that caused the demon to be fired.

The resource models have been encoded as a-lists that are attached directly to the scroll table. The functions that make an assignment in response to a request, or that enter data describing a current or expected event, do so by referencing the a-list for the entry type and using it to control the actual entry operation.

The use of demons is a very important technique in the implementation of ACS.1. They are used to enforce consistency according to the resource model, while maintaining separation of the operations involved. For example, the use of demons allows the data entry function to be separated from its side effects.

As a specific example, suppose a given pilot goes on sick leave for some period of time. That information is entered directly. The function that makes the entry into the scroll table does so without regard for any possible side-effects. If that entry overlaps a period during which the pilot is assigned to a mission, this fact is recognized by a demon that has been set to watch the ASG entries. The demon recognizes that an assignment must be made for the full duration or not at all. The demon therefore cancels the assignment over the full range. It then reschedules the mission with a new pilot if it has been authorized to do so and if one is available. If not authorized, or if no other pilot is available, it alerts the commander, calling a dialog function through which he can evaluate the situation and decide what to do. The demon is the link between the entry and its side effects, however extensive these may be.

----

* A "demon", as the term is used in the artificial intelligence community, is a function that is set to "watch" a data element. If the data changes in a way that is specified as part of the demon, the function is "fired" and does what it has been programmed to do.
As used here, demon refers only to "write demons." There are also "read demons" that can be fired on reading the data element, but we have had no occasion to use them in ACS.1.

We give this rather detailed example to illustrate how demons are used to maintain consistency in the scroll table according to the resource model. Other examples could be given where different rules are involved. As a general principle, whenever a change of the contents of some cell implies a change in the restrictions on the data elsewhere, demons provide a convenient device for testing the new restrictions and making any necessary adjustments.

The commander has control over what the demon should do when it is fired. In the example discussed, he may or may not be willing to have the system reschedule the mission with a different pilot. To switch from one mode to the other, it is only necessary to change the function called by the demon. This function is named in the resource model, and a new function name can be substituted without difficulty. If the alternate function does not exist in the system, an appropriate one can be constructed quickly since it need only implement the specific policy decision without considering other problems. It is, therefore, relatively easy for the user to exert effective control over what authority is exercised by the system.

The effect of the use of demons is to separate the inputting of data from its possible side effects. The entry function need only check the admissability of the data, and, if permitted, enter it. Any consequent adjustment or deletion of other data is handled by demons.

Demons are also used for other purposes. For example, as was discussed, the commander may need to maintain a certain number of pilots available for missions during some critical period. Demons are used to monitor the data for this purpose, issuing an alert when the condition is no longer met. These demons are set to watch the columns of the scroll table. Any change in the data in those columns fires the demon, which then evaluates the reserve capability that remains. If the reserve has fallen below the required limit, an alert message is initiated.

Demons can also be used on a row of a scroll table, to be fired when the data in the given row changes in specified ways. This type of demon can be used to accumulate flight hours per pilot for the month, or to keep track of when scheduled maintenance actions should be undertaken on the aircraft and, if desired, initiate the planning for those actions.

The use of demons is a flexible and powerful tool for applying the knowledge contained in the resource model of the scheduler and for obtaining alert messages. It is an important device for achieving separation of functional behavior, as illustrated in the separation of of data entry from its side effects and from the determination of critical conditions or of cumulative measures. This separation is important since it means that changes in the rules and policies contained in the resource model can be made with minimal concern for their long range implications.

The design of the schedulers, and the means used to implement them, is intended to give the user immediate and easy control over the knowledge they use, and the way they use that knowledge. It represents a deliberate exploitation of the principle of separation, seeking to isolate, as far as possible, the different aspects of scheduler operations.

VI. CONCLUSIONS

The experience with ACS.1 has demonstrated the feasibility of a system that can be a direct aid to management in planning, executing, and monitoring operations. The system is a knowledge-based, model-driven one that exhibits considerable inferential capability, yet is controllable by the manager in a flexible and adaptable way.

Flexibility and adaptability are considered to be of prime importance so that the system can continue to be responsive to managerial needs as the situation develops, as the organization and its policies evolve, and as managerial requirements change. Flexibility and adaptability are obtained through several means that can be regarded as different aspects of the principle of separation. The primary applications of the principle are the following:

* The system, as seen by the user, is highly modular, and its structure parallels that of the comparable human organization. In consequence, operations can be shifted between ACS.1 and a human at will, with little difficulty.

* A central node is provided in the unit called the message handler. This node provides a convenient means for the user to intervene to direct operations or to take over, permanently or temporarily, control of any of the system functions.

* The virtual modules at the top level, the planners and schedulers, are entirely separated in their operations, being linked only through the message handler. One module does not even know if another exists, or if its functions are being handled by a human at a terminal.

* The different functions of ACS.1--planning, administering approved plans, monitoring their execution, and retrospective analysis of past operations--are well differentiated. Consequently, each can be modified or adapted independently of the others.

* Both the process models and resource models used by ACS.1 are explicitly encoded in the system. Since these models embody the knowledge-base of the system, the knowledge is accessible for modification, adaptation, and evolution.

* In the schedulers, extensive use is made of demons to separate data entry from its side effects. This greatly facilitates modifying scheduler operations, and creating new schedulers at will.

* Demons are used for other purposes such as monitoring for overloading or underloading of a given type of resource, accumulating data, or initiating actions whose timing depends on accumulated indices. Again, they provide separation of functions from the other operations of the system.

The system demonstrates the possibility of achieving the desired attributes and behavior for a management support system. In particular, it is a system in which the user can exert a very high degree of control with minimal attention to implementation details. It provides him with a flexible means for evaluating the expected state of the environment. It also provides him with a facility for automaticly monitoring against exceptional conditions. Further, the system acts to enforce the consistency of the data according to the knowledge it has about the processes and resources that are its concern. Finally, it can generate plans, and administer and maintain those plans in a changing environment. The degree of autonomy it exercises in these activities is under the control of the manager, so that he retains full responsibility.

It is believed that the ACS.1 system confirms the possibility of making knowledge-based inferential systems useful in the decision-making role of the manager.

REFERENCES

1. C. Green, "The Application of Theorem-Proving to Question-Answering Systems," Doctoral Dissertation, Elec. Eng. Dept., Stanford University, Stanford, CA, June 1969. Also printed as Stanford Artificial Intelligence Project Memo AI-96 (June 1969).

2. R. Quillian, "The Teachable Language Comprehender: A Simulation Program and Theory of Language," Comm. ACM, Vol. 12, pp.469-476 (1969).

3. R. E. Fikes and N. J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence, Vol. 2, No. 3-4, pp 189-208 (Winter 1971).

4. C. Hewitt, "Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot," Doctoral Thesis, Dept. of Math., Mass. Inst. of Tech., Cambridge, MA (1972).

5. G. J. Sussman and T. Winograd, "MICRO-PLANNER Reference Manual," MIT Artificial Intelligence Laboratory, Memo No. 203, Cambridge, MA (July 1970).

6. D. V. McDermott and G. J. Sussman, "The CONNIVER Reference Manual," MIT Artificial Intelligence Lab., Memo No. 269, Cambridge, MA (May 1972).

7. T. Winograd, "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language," MIT Artificial Intelligence Lab., Tech. Report AI-TR-17, Cambridge, MA (1971). Also published as "Understanding Natural Language," (Academic Press, New York, 1972).

8. E. D. Sacerdoti, "Planning in a Hierarchy of Abstraction Spaces," Proc. International Joint Conference on Artificial Intelligence (August 1973).

9. M. Minsky, "A Framework for Representing Knowledge," MIT Artificial Intelligence Lab., Memo No. 306, Cambridge, MA (June 1974).

10. T. Winograd, "Five Lectures on Artificial Intelligence," Stanford Artificial Intelligence Laboratory, Memo No. AIM-246, Stanford, CA (September 1974).

11. G. A. Korn and T. M. Korn, "Mathematical Handbook for Scientists and Engineers," McGraw-Hill Book Co., Inc., New York, 1961, p. 12.6-1.

12. "Computer and Job-Shop Scheduling Theory," Ed. by E. G. Coffman, Jr. Ed. (John Wiley & Sons, New York, 1976).

DISTRIBUTION LIST

Defense Documentation Center Cameron Station Alexandria, Virginia 22314	12 copies
Office of Naval Research Information Systems Program Code 437 Arlington, Virginia 22217	2 copies
Office of Naval Research Code 102IP Arlington, Virginia 22217	6 copies
Office of Naval Research Branch Office, Boston 495 Summer Street Boston, Massachusetts 02210	1 copy
Office of Naval Research Branch Office, Chicago 536 South Clark Street Chicago, Illinois 60605	1 copy
Office of Naval Research Branch Office, Pasadena 1030 East Green Street Pasadena, California 91106	1 copy
New York Area Office 715 Broadway - 5th Floor New York, New York 10003	1 copy
Naval Research Laboratory Technical Information Division, Code 2627 Washington, D.C. 20375	6 copies
Dr. A. L. Slafkosky Scientific Advisor Commandant of the Marine Corps (Code RD-1) Washington, D.C. 20380	1 copy
Office of Naval Research Code 455 Arlington, Virginia 22217	1 copy
Office of Naval Research Code 458 Arlington, Virginia 22217	1 copy
Naval Electronics Laboratory Center Advanced Software Technology Division Code 5200 San Diego, California 92152	1 copy

Mr. E. H. Gleissner Naval Ship Research & Development Center Computation and Mathematics Department Bethesda, Maryland 20084	1 copy
Captain Grace M. Hopper NAICOM/MIS Planning Branch (OP-916D) Office of Chief of Naval Operations Washington, D.C. 20350	1 copy
Mr. Kin B. Thompson Technical Director Information Systems Division (OP-91T) Office of Chief of Naval Operations Washington, D.C. 20350	1 copy
Director National Security Agency Attn: Mr. Glick Fort George G. Meade, Maryland 20755	1 copy
Naval Aviation Integrated Logistic Support Center Code 800 Patuxent River, Maryland 20670	1 copy
Professor Omar Wing Columbia University in the City of New York Department of Electrical Engineering and Computer Science New York, New York 10027	1 copy
Mr. M. Culpepper Code 183 Naval Ship Research and Development Center Bethesda, Maryland 20084	1 copy
Mr. D. Jefferson Code 188 Naval Ship Research and Development Center Bethesda, Maryland 20084	1 copy
Robert C. Kolb, Head Tactical Command Control and Naval Division Naval Electronics Laboratory Center San Diego, California 92152	1 copy